Python: module regrid.crossSection

regrid.crossSection

index

Modules

MA
Numeric
regrid._regrid
copy
string

Classes

CrossSectionRegridder

class ***CrossSectionRegridder***

```
    #-----------------------------------------------------------------
    #
    #    PURPOSE: To perform all the tasks required to regrid the input
    #             latitude-level plane for all times
    #
    #    PROCEDURE: Step One:
    #                  Make an instance of class CrossSectionRegridder
    #               Step Two:
    #                  Pass the input data with some descriptive parame
    #                  in return
    #
    #-----------------------------------------------------------------
```

Methods defined here:

__call__(self, ar, missing=None, order=None, method='log')

```
      Call the regridder function.
      ar is the input array.
      missing is the missing data value, if any. It defaults to the
        defined for the input array, if any.
      order is of the form "tzyx", "tyx", etc.
      method is either 'log' to interpolate in the log of pressure,
```

__init__(self, latIn, latOut, levIn, levOut, latTypeIn=None, latSizeIn=None, latTypeOut=None, latS

```
      #-----------------------------------------------------------
      #
      #    PURPOSE: To make an instance which entails setting up th
      #
```

```
#     DEFINITION:
#
#              def __init__(self, latIn, latOut, levIn, levOut
#                                                          latType
#
#     PROCEDURE:
#
#         The user must assemble at least the following four p
#
#             latIn -  the axis specifying the latitude grid f
#
#             latOut -  the axis specifying the latitude grid
#
#             levIn -  the axis specifying the pressure grid f
#
#             levOut -  the axis specifying the pressure grid
#
#
#         Additional information is required if a latitude gri
#         Otherwise it is a subset of one of the standard glob
#         for the grid type must be 'gaussian', 'equalarea', '
#         computation requires the size of the global grid fro
#         the user must assemble:
#
#               latTypeIn -- for input latitude, one of the fo
#                                'gaussian'
#                                'equalarea'
#                                'uniform'
#                                'generic'
#
#               latSizeIn -- for input latitude, the size of t
#
#               latTypeOut -- for output latitude, one of the
#                                'gaussian'
#                                'equalarea'
#                                'uniform'
#                                'generic'
#
#               latSizeOut  -- for output latitude, the size o
#
#     USAGE:
#
#           To  make an instance preparing for a global to glo
#
#              r = CrossSectionRegridder(latIn, latOut, levIn,
#
#           To  make an instance preparing for a global to a r
#           a global gaussian grid of size 64, type
#
#              r = CrossSectionRegridder(latIn, latOut, levIn,
#
#           where the latOut axis must have been selected from
```

```
        #-------------------------------------------------------------
```

rgrd(self, dataIn, missingValueIn, missingMatch, logYes='yes', positionIn=None, maskIn=None, m

```
        #-------------------------------------------------------------
        #
        #    PURPOSE: To perform all the tasks required to regrid the
        #             the latitude-level plane.
        #
        #    DEFINITION:
        #
        #             def rgrd(self, dataIn, missingValueIn, missingM
        #
        #
        #
        #    PASSED :  dataIn -- data to regrid
        #
        #             missingValueIn -- the missing data value to use
        #                               and there are two choices:
        #                                     None -- there is no mis
        #                                     A number -- the value t
        #                               The presence of missing data
        #
        #             missingMatch -- the comparison scheme used in s
        #                              in as missingValueIn. The choic
        #                                   None -- used if None is th
        #                                   exact -- used if missingVa
        #                                   greater -- the missing dat
        #                                   less -- the missing data v
        #
        #             logYes -- choose the level regrid as linear in
        #                       'yes' for log. Anything else is linea
        #
        #
        #              positionIn -- a tuple with the numerical posit
        #                            in C or Python order specified i
        #                            level and time. Latitude and lev
        #                            slot in the tuple. Notice that t
        #                            always three.
        #
        #                            Explicitly, in terms of the shap
        #
        #                                 positionIn[0] contains the
        #                                 positionIn[1] contains the
        #                                 positionIn[2] contains the
        #
        #                            As  examples:
        #                                 If the c order shape of 3D
        #                                     (number of times, numbe
        #                                 submit
        #                                      (2, 1, 0).
        #
        #                                 If the c order shape of 2D
```

```
#                                       (number of times, numbe
#                                   submit
#                                       (1, None, 0).
#
#                               Send in None if the shape is a s
#                               as follows:
#                                  2D -- code assumes (1,0,None)
#                                  3D -- code assumes (2,1,0)
#
#             maskIn -- an array of 1.0 and 0.0 values where
#                       mask only works on the latitude grid
#                       plane. The 0.0 value removes the dat
#                       following choices:
#
#                       None -- an array of 1.0s is created
#                               data in the input data array
#
#                       array -- an array of 1.0s or 0.0s wh
#                                dataIn. This user supplied
#                                required to account for mis
#                                and missingMatch to supply
#                                data array, dataIn.
#
#
#             missingValueOut -- the value for the missing d
#                                default entry, None, the co
#                                1.0e20
#
#
#    RETURNED : dataOut -- the regridded data
#
#
#    USAGE:
#
#          Example 1.  To regrid dataIn into dataOut using al
#                      missing data.
#              dataOut = x.rgrd(dataIn, None, None)
#
#          Example 2.  To regrid dataIn into dataOut using 1.
#
#                      dataOut = x.rgrd(dataIn, 1.e20, 'great
#
#    WARNING: This code does not regrid cross sections which
#
#
#-----------------------------------------------------------------
```

Functions

checkdimension(x, name)

```
#-----------------------------------------------------------------
#
#    purpose:  dimension  checks
#                  1. has a len method
#                  2. data type is float32
#                  3. monotonically increasing vectors
#
#    passed :  x - coordinate vector
#              name - coordinate vector ID
#
#    returned: x, xsize -- dimension vector and its size
#
#-----------------------------------------------------------------
```

get_latitude_wts_bnds(checklatpass)

```
#-----------------------------------------------------------------
#
#     routine: get_latitude_wts_bnds
#
#     purpose: compare the passed checklatpass with the correct ge
#              ones calculated here. After finding a match call th
#              to get the bounds.
#
#     usage:   wts,bnds = get_latitude_wts_bnds(checklatpass)
#              where checklatpass is the grid to check
#
#    return:   wts, bnds - tuple with weights and bounds
#
#-----------------------------------------------------------------
```

get_region_latitude_wts_bnds(latRegionpass, latType, latSize)

```
#-----------------------------------------------------------------
#
#     routine: get_region_latitude_wts_bnds
#
#     purpose: compare the passed latitudes, latRegion, with the g
#              ones calculated here and extract the wts and bounds
#              the region
#
#     usage:   wts,bnds = get_region_latitude_wts_bnds(latRegion,
#              where latRegion is the regional grid to check
#
#    return:   wts, bnds - tuple with weights and bounds
#
#-----------------------------------------------------------------
```

latitude_bounds(lat_bnds)

```
#-----------------------------------------------------------------
#
#     purpose: set up the shape and bounds for use by maparea
#
#     usage:
```

```
#
#     returned:  tuple ( bn,bs )
#
#--------------------------------------------------------------------
```

rmserror(data1, data2)

```
#--------------------------------------------------------------------
#
#    purpose: compute the rms error for two data sets having the s
#
#    passed : the two data sets
#
#    returned: rms error
#
#--------------------------------------------------------------------
```

section(latvals, levvals)

```
#--------------------------------------------------------------------
#
#    purpose: make the crossi section analytical test case
#
#    passed :  the grid coordinate vectors
#
#    returned: xsection -- a temerature like cross section
#
#--------------------------------------------------------------------
```

sectionmask(dataIn, positionIn, maskIn, missingValueIn, missingMatch)

```
#--------------------------------------------------------------------
#
#     purpose: construct the mask for the input data for use by rg
#
#     usage:   amskin = mask(dataIn, positionIn, maskIn, missingVa
#
#     returned: amskin
#
#--------------------------------------------------------------------
```

sendmsg(msg, value1=None, value2=None)

```
#--------------------------------------------------------------------
#
#    purpose: send the same message to the screen
#
#    passed :  msg - the string
#              value - the number associated with the string
#
#    returned: return
#
#--------------------------------------------------------------------
```